【H-debugger 対応】

TOPPERS/OSEK カーネル

アプリケーションノート

Ｈ８／３００Ｈ　Ｔｉｎｙシリーズ（Ｅ８）
Ｈ８／３６８７Ｆ　ＫＰＩＴ－ＧＮＵ［Ｈｅｗ］版
（マイコンカーラリー用のハードを使用）

２００８／１０／０２

Ｒｅｖ１．５０（２００８／１０／０２）

エーワン株式会社

目　　　次

エーワン株式会社

第１章　概要

１－１．はじめに

本アプリケーションノートは、TOPPERS/OSEK カーネルをもとに「H-debugger」と「H8/3687F」用にポーティングしたサンプルソフトです。
サンプルソフトの他アプリケーションへの利用／変更に関しての制限は一切ありませんので自由にお使い下さい。ただし、このサンプルソフトの不具合により発生した損害に対しての責任、及び、修正の義務は負いません。また、このサンプルソフトに関する質問の回答義務も負えませんが、メールでのお問い合わせに関しては、弊社責任の範囲内でしたら出来るだけ御答えするように努めます。Mail: cat-i@aone.co.jp

１－２．関連文書

本アプリケーションは、下記ドキュメントを参考にして作成しました。

１）TOPPERS/OSEK カーネル外部仕様書 ――――――――――――　株式会社ヴィッツ製
２）TOPPERS/OSEK カーネル SG 取扱説明書　――――――――――　株式会社ヴィッツ製
３）TOPPERS/OSEK カーネルアプリケーションノート　――――　株式会社ヴィッツ製
４）OSEK/VDX　Operating System Ver2.2.1　――――――――　OSEK/VDX 仕様
５）OSEK/VDX　Binding Specification Ver1.4.2 ――――――　OSEK/VDX 仕様
６）OSEK/VDX　OIL Specification Ver2.5　――――――――　OSEK/VDX 仕様

＊OSEK/VDX が公開している仕様書は、http://www.osek-vdx.org/ よりダウンロードにより入手して下さい。

１－３．開発環境

１）Hew Version 4.04.01.001――――――――――――――――――Renesas 製
２）KPIT GNUH8[ELF] Toolchain v0801 ――――――――――――KPIT 製
にて作成しましたので各自用意をして下さい。

１－４．製品梱包内容

１）サンプルソフト用 CD（本書 PDF ファイルも含む）――――　１枚
⚠　・本製品の価格体系は、CD 配布の実費のみになっております。

１－５．サポート

TOPPERS/OSEK に関するサポートが必要な場合は、TOPPERS ホームページの「関連製品」「サポート・サービス」の項をご覧になり、御利用下さい。
http://www.toppers.jp/　<――　TOPPERS ホームページ

第２章　TOPPERS/OSEK カーネルの開発階層

```
¥toppers_osek
        ¥config                                    // 機種依存階層
                ¥h8t-kpitgnu-3687                  // 開発環境分類
                        +CPU 依存部
                        ¥H3687                     // システム依存部

        ¥include                                   // インクルードファイル階層
        ¥kernel                                    // カーネル共通部階層
        ¥sample                                    // サンプル階層(未使用)
        ¥sg                                        // システムジェネレータ階層
                ¥impl_oil                          // OIL 記述の実装定義部階層

        ¥syslib                                    // システムライブラリ階層
                ¥h8t-kpitgnu-3687                  // 開発環境分類
                        +CPU 依存部
                        ¥H3687                     // システム依存部

        ¥tools                                     // Hew 管理階層
                ¥h8t-kpitgnu-3687                  // 開発環境分類
                        +ワークスペース
                        ¥H3687                     // プロジェクト１
                                ¥appsrc            // サンプルソース [*.c]
                                ¥debug             // オブジェクト   [*.x]
                        ¥Project                   // 新規プロジェクト作成用
                                                   // テンプレート
```

本サンプルソフトは、１本のプロジェクトを用意しています。

１）プロジェクト１　　¥H3687　　　実行オブジェクト【H3687.x/H3687.mot】

　　本アプリケーションノートのサンプルプロジェクト

青字部分は、TOPPERS/OSEK 正式リリース（オープンソース）になります。その他は弊社で改造および作成をしました。

第３章　サンプルアプリケーションの概要(H3687)

３－１.　 サンプルアプリケーションの構成

サンプルアプリケーションは、下記の構成にて作成しました。

１）５個のタスク

- ① MainTask ------------------ 各 Task の起動処理
- ② Task1 ------------------------ センサー入力(8 点)の状態表示
- ③ Task2 ------------------------ PWM1/2/3 の 3 点パルス出力
- ④ Task3 ------------------------ RS232C による PWM デューティー比変更
- ⑤ HighPriorityTask -------- 各タスクのスタックオーバーバー監視

２）2 個の割込み処理

- ① システムタイマー　(TimerB1)1ms 割込み　SysTimerInt()
- ② SCI_1　Rx/Tx/Err　RxTxErrHwSerialInt()

３）５個のイベント

MainEvt/T1Evt/T2Evt/T3Evt/THEvt

４）２個のアラーム

MainCycArm/SetEvtHArm

５）２個のコールバック処理

CallBackArm(TimerCallBack コールバック利用のソフトタイマ)

６）１個のウォッチドッグタイマー

WatchDogClear

７）スタートアップフックルーチン

ErrorHook

８）シャットダウンフックルーチン

ShutdownHook

９）プレタスク／エラーフックルーチン

PreTaskHook/ErrorHook

１０）1 個のアプリケーションモード

AppMode1

３－２．　動作フロー（ゼネラル）

PRIORITY = 14

MainTask

イベント待　　5ms 毎のサイクル起動　WaitEvent(MainEnv)

全 Task の StatusGet　　全 Task の状態を取得　GetTaskState(....)

DIPSW 入力処理　　Task 起動の為、DIPSW 入力　SwInput()

DIPSW 入力　　DIPSW 状態入力 Sw = GetInSw()

Task1 起動?　　Yes　Task1 起動 ActTask() イベント起動

Task1 Wait　　Yes　Task1 にイベント発行 SetEvent(..)

Task2 起動?　　Yes　Task2 起動 ActTask() イベント起動

Task2 Wait　　Yes　Task2 にイベント発行 SetEvent(..)

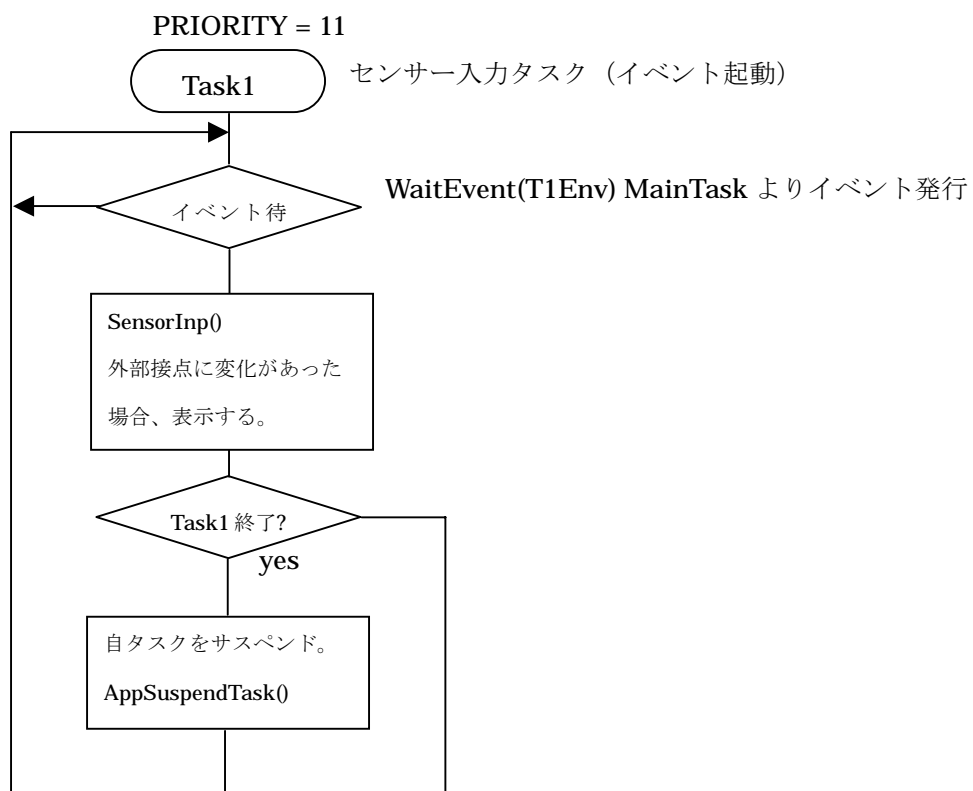
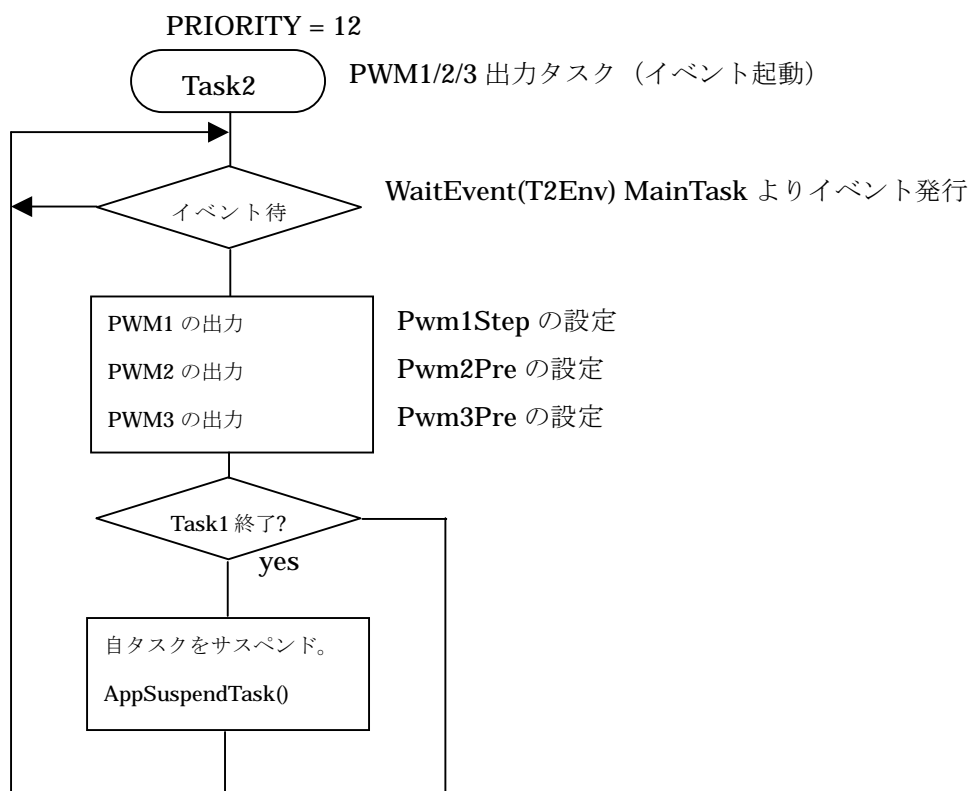
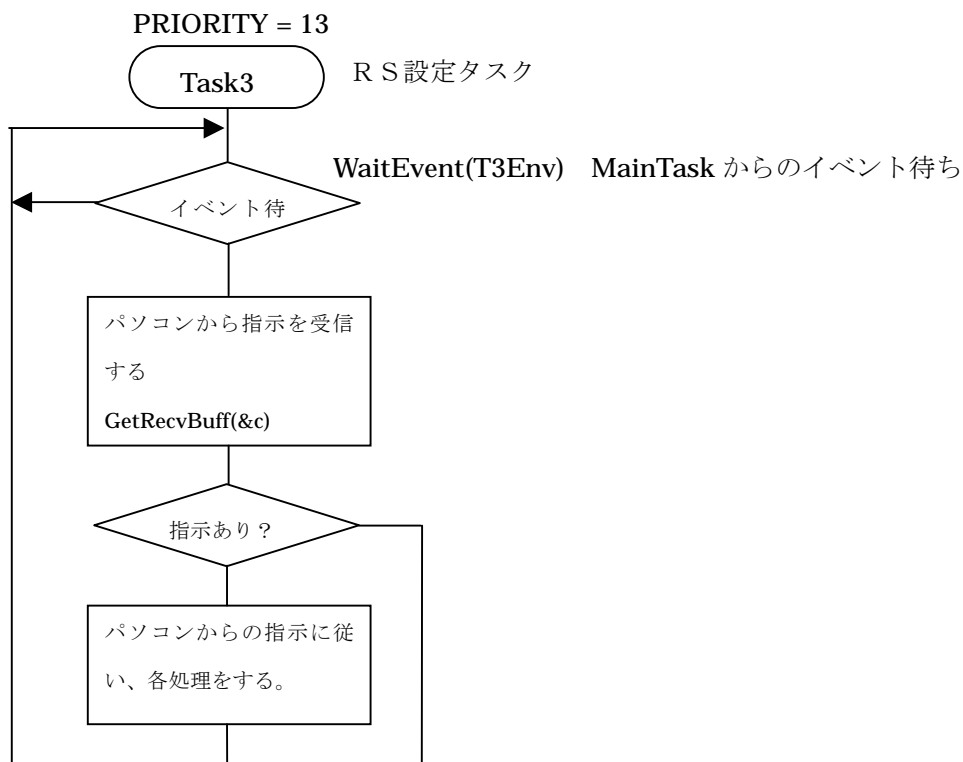
Task3 起動?　　Yes　Task3 起動 ActTask() イベント起動

Task3 Wait　　Yes　Task3 にイベント発行 SetEvent(..)

動作状態を LED で表現する。

①Task1(センサー入力)の起動条件　－－　DIPSW(1)ON で起動する。

②Task2(ＰWM出力)の起動条件　－－－　PushSW が ON

③Task3(ＲＳ設定)の起動条件　－－－－　パソコンよりスペース入力

①Task1 の終了条件　ーー　DIPSW(1)でなくなった時、終了とする。

①Task2 の終了条件　――　PushSW の ON で終了とする。

＜パソコンからの指示コード＞

① 'Q'キー　－－－－－－－ PWM1 センタ+1　(-50～ 50)
② 'W'キー　－－－－－－－ PWM1 センタ-1　(-50～ 50)
③ 'A'キー　－－－－－－－ PWM2 パーセント+1　(-100～100)
④ 'S'キー　－－－－－－－ PWM2 パーセント-1　(-100～100)
⑤ 'Z'キー　－－－－－－－ PWM3 パーセント+1　(-100～100)
⑥ 'X'キー　－－－－－－－ PWM3 パーセント-1　(-100～100)
⑦ Space　－－－－－－－ メニュー再表示
⑧ 'E'キー　－－－－－－－ Task3 の終了指示

PRIORITY = 15

HighPriorityTask

各 Task のスタック監視タスク

イベント待

10ms 毎のサイクル起動　WaitEvent(THEnv)

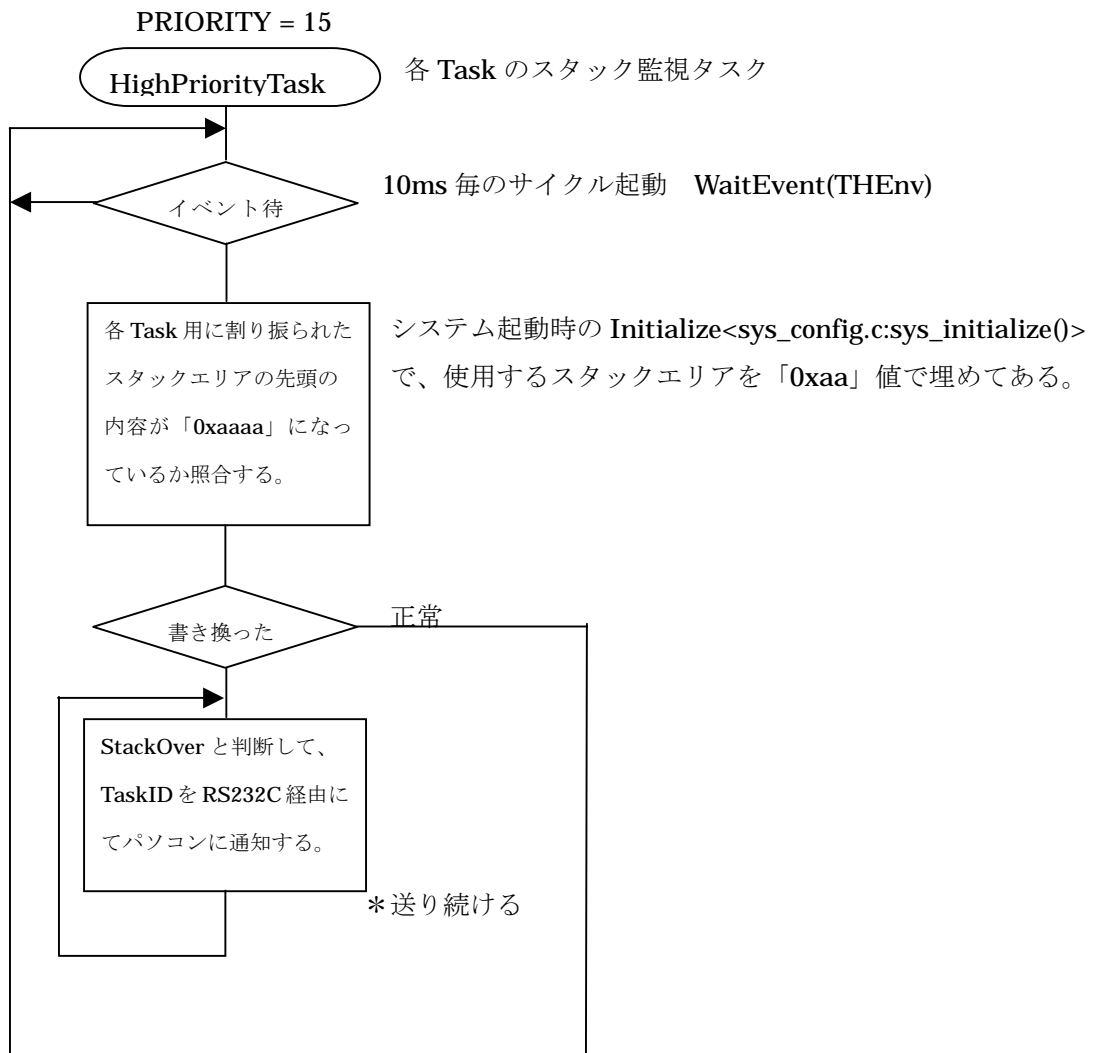
各 Task 用に割り振られたスタックエリアの先頭の内容が「0xaaaa」になっているか照合する。

システム起動時の Initialize<sys_config.c:sys_initialize()>で、使用するスタックエリアを「0xaa」値で埋めてある。

書き換った

正常

StackOver と判断して、TaskID を RS232C 経由にてパソコンに通知する。

＊送り続ける

３－３． H-debugger 対応に必要なポーティング

＜デバッグモードがエミュレーションの場合＞

H8S/3687F は E8 仕様の為、デバッガ用ポーティングは不要です。

＜デバッグモードがユーザモードの場合＞

1）ポーティング手続き

```
.¥config¥h8t-kpitgnu-3687¥H3687¥sys_support.src

hardware_init_hook:
        ldc.b   #H'80, ccr

        ; H-debugger 用 リセット遅延の為、20msWait
        ; Target H8/3687-14.7456MHz
wait20ms:
        mov.w   #20,r0
waitrst:
        bsr     _SOFT1MS
        dec.w   #1,r0
        bne     waitrst
        rts
;/*SOFT1MS()  1ms ｿﾌﾄﾀｲﾏｰ (14.7456MHz)  Non Wait
_SOFT1MS:
        push.w  r0              ;
        mov.w   #1843,r0        ;/* 1843*8=14744cyc  */
wait:                                   ;
        dec.w   #1,r0           ;/* 2 clock         */
        nop                     ;/* 2 clock         */
        bne     wait:16         ;/* 4 clock         */
        pop.w   r0              ;
        rts
```

＜理由＞

ユーザモードでのデバッガ処理は、RESET 立ち上げ後、NMI 割込みをターゲットに対して掛け、デバッガ用モニタを起動させています。

その NMI が掛かるまでに「main()」に飛ばないように、ここでループさせ止めます。

（DEF－CPU 設定にて 200ms 遅延回路を使用しない場合の例です）

３－４． H-debugger でのプロファイル表示(DEF 7.00B 以上)

ターゲット側の RAM を使用して、各 Task のプロファイル表示をします。

１）ポーティング手続き

```
.¥syslib¥h8t-kpitgnu-3687¥sys_timer.h


#define USE_PROFILE
#ifdef   USE_PROFILE
        #define   PROFILESIZE              512
        #define   PROFILEPRETASK           1
#endif
```

①「USE_PROFILE」を有効にすると、プロファイル表示が可能になります。不要になった場合は、コメントアウトして下さい。

②「PROFILESIZE」は、ターゲット側で確保する RAM のバイト数になります。アプリケーションに応じて調整して下さい。

③「USE_PROFILE」を有効にする事により、下記変数が確保されます。

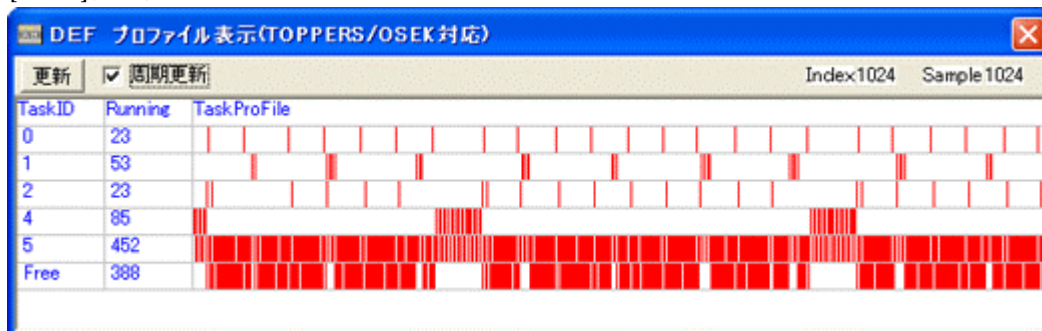
```
UINT8          _TaskProFile[PROFILESIZE];
UINT8          TaskProFlg;
UINT16         TaskProFileIdx;
```

④システムタイマー割り込みのタイミングで現 Running 中の TaskID を「_TaskProFile」に順次記憶していきます。(リングバッファ)

⑤サンプリング場所は、「プレタスクフックルーチン」と「ISR(SysTimerInt)」の 2 箇所です。

⑥プロファイル表示させたい場合は、

DEF メニュー<データ>－<プロファイル表示>をクリックして下さい。

[3-4-1]表示例

「更新」PB をクリックしますと最新データを表示します。

「周期更新」にチェックしますと、常時最新状態をターゲットからオンザフライ機能により、データを収集し、Task 状態を表示します。

第４章　ハード構成およびシステム構成

４－１．　ハード構成

この解説書を進めるにあたり、下記ハード構成の準備をお願いします。

センサー基板(8 点)
0 1 2 3 5 6 7

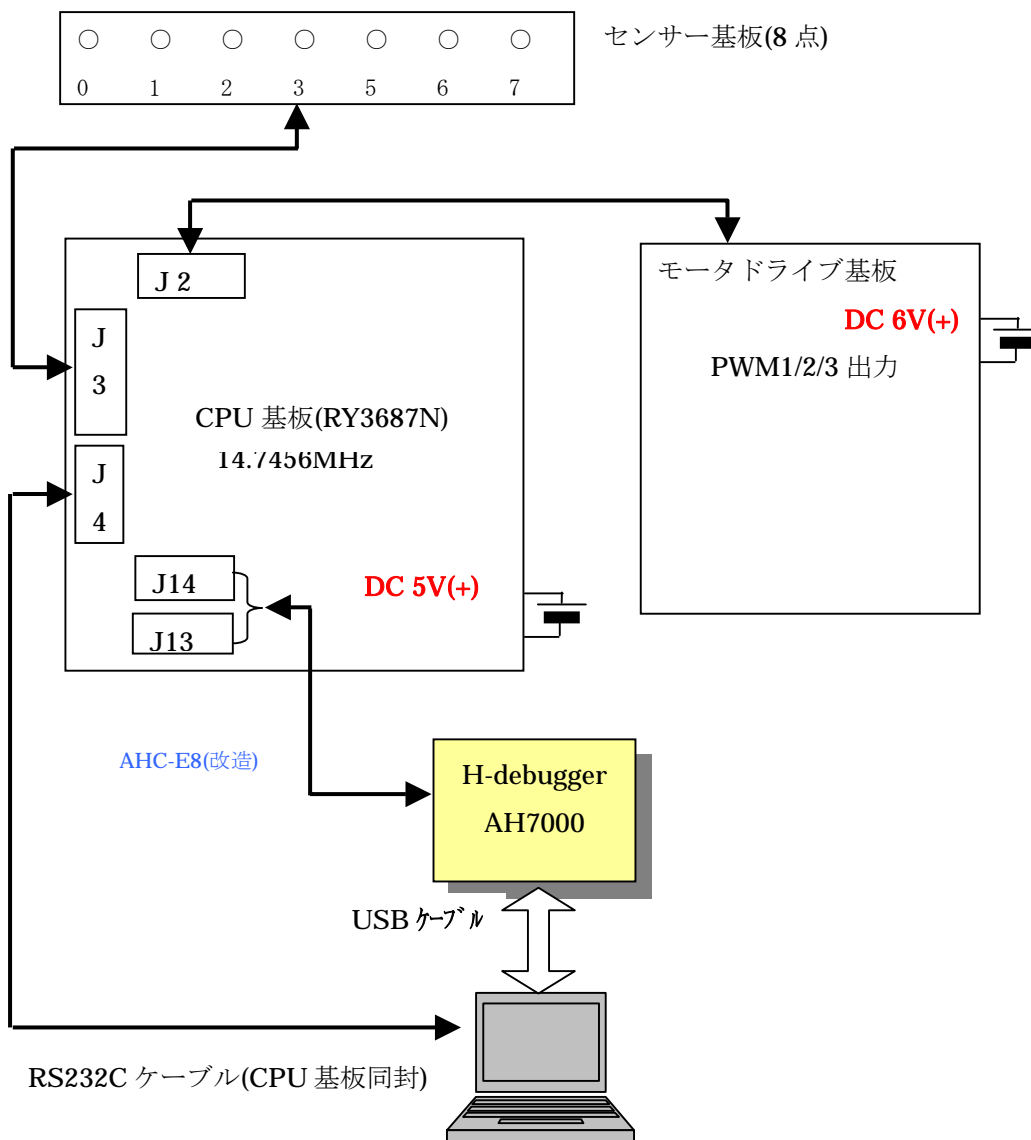
モータドライブ基板
DC 6V(+)
PWM1/2/3 出力

J 2
J 3
J 4
CPU 基板(RY3687N)
14.7456MHz
J14
J13
DC 5V(+)

AHC-E8(改造)

H-debugger
AH7000

USB ｹｰﾌﾞﾙ

RS232C ケーブル(CPU 基板同封)

＜デバッグに必要なオプションケーブル＞

①AHC-E8(改造)　　　AHC-E8（標準品）を RY3687N 用に改造　　1 本

1）CPU 基板(RY3687N)の J3 とセンサ基板をケーブル接続します。

2）CPU 基板(RY3687N)の J13/J14 と AH7000 をケーブル(AHC-E8 改造)で接続します。

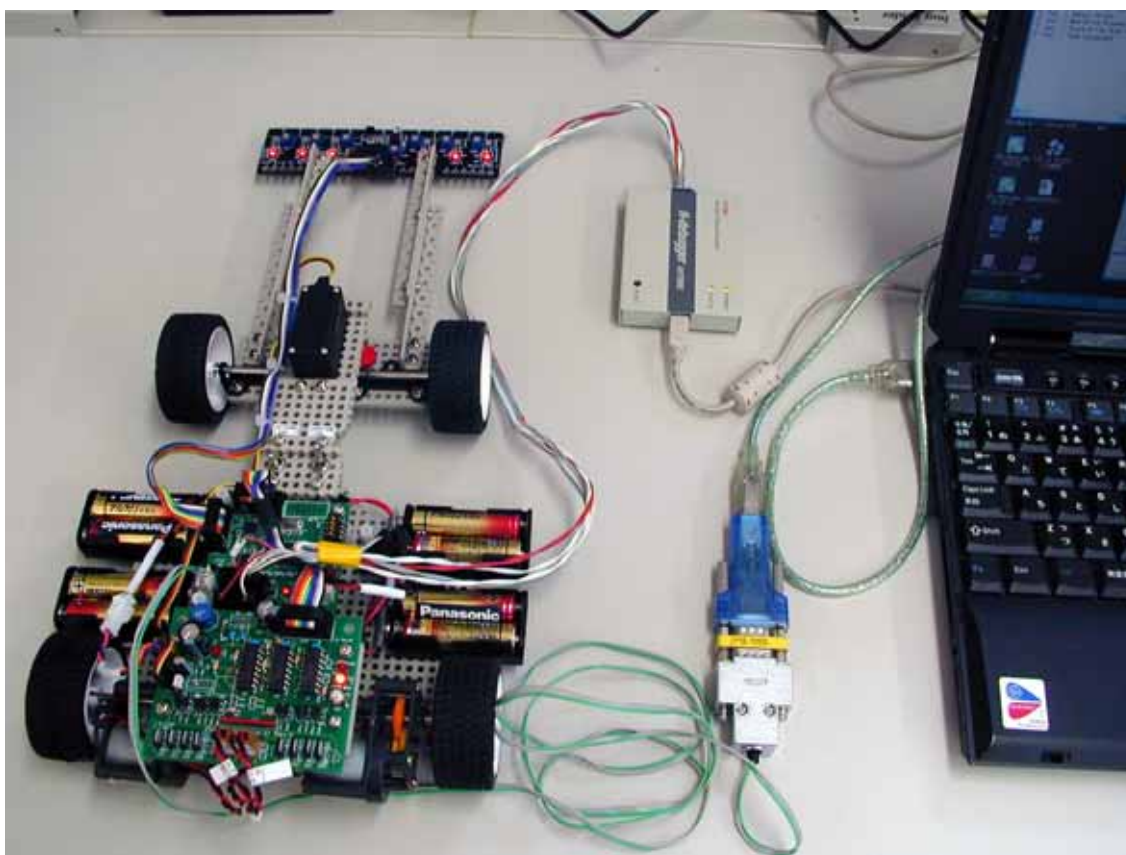
3）AH7000 とパソコンを USB ケーブルで接続します。

4）CPU 基板(RY3687N)の J2 とモータドライブ基板をケーブル接続します。

5）CPU 基板(RY3687N)の J4 とパソコンを RS232C ケーブル(同封品)で接続します。

6）CPU 基板(RY3687N)に電源（DC+5V)が供給できるように接続します。

7）モータドライブ基板に電源（DC+6V)が供給できるように接続します。

［接続例］

４－２．システム構成

サンプルのシステムブロック図およびメモリマップと I/O 表を記述します。

４－２－１．H8/3687F のプログラムメモリ MAP

メモリ	アドレス
CPU 内部 H8/3687F フラッシュ ROM 56Kbyte	0x0--------------ベクタテーブル① 0x200-----------モニタ（ファーム）エリア 0x800-----------アプリケーションプログラム開始番地② 0xDFFF-------FROM 最終
CPU 内部 RAM2 2Kbyte	0xE800--------アプリケーション使用 RAM2 の開始番地③ 0xEFFF-------RAM2 の終了番地
CPU 内部 RAM1 2Kbyte	0xF780---------デバッグモニタが使用する RAM 開始番地 0xF7C0--------カーネル使用 RAM の開始番地④ 0xFF7F-------初期スタックポインタ位置(0xFF80)⑤

＜セクション名>

① ベクタテーブル-- .vects

②アプリケーションプログラムの開始番地---------- .text,.init,.rodata....etc

③アプリケーション使用 RAM2 の開始番地---------.bss2

④カーネル使用 RAM の開始番地---------------------- .data,.bss....etc

⑤初期スタックポインタ位置--------------------------- .stack

４－２－２．プログラムサイズの詳細 MAP

開始番地	サイズ	分類
0x200	0x600	デバッガモニタ（ファーム）
0x800	0xC54	アプリケーション（main....etc）
0x1454	0x10DC	割込みハンドラ、内部 I/O 関係の関数
0x2530	0x2288	TOPPERS/OSEK カーネル
0x47B8	0x4C7	C ライブラリ、ROM テーブル、etc
0x4C7F		最終アドレス

４－２－３．CPU 基板(RY3687N)の I/O マップ表

CPU 基板の DIP-SW					
	ﾎﾟｰﾄｼﾝﾎﾞﾙ CPU 基板			方向	信号名
PDR3	P33			入力	DIP-SW4(0=ON)
	P32			”	DIP-SW3(0=ON)
	P31			”	DIP-SW2(0=ON)
	P30			”	DIP-SW1(0=ON)

センサー関係　J3					
	ﾎﾟｰﾄｼﾝﾎﾞﾙ CPU 基板	ピン番号 CPU 基板		方向	信号名
PDR5	VCC	1		VCC	+5V
	P57	2		入力	センサ 7
	P56	3		”	センサ 6
	P55	4		”	センサ 5
	P54	5		”	センサ 4(スタートバー)
	P53	6		”	センサ 3
	P52	7		”	センサ 2
	P51	8		”	センサ 1
	P50	9		”	センサ 0
	GND	10		GND	GND

RS232C 関係　J4　　適用ケーブル　CPU 基板（同封品）					
	ﾎﾟｰﾄｼﾝﾎﾞﾙ CPU 基板	ピン番号 CPU 基板		方向	信号名
	P22/TXD	1	TXD	出力	CH1-SCI3(RS レベル)
	GND	2	GND	GND	GND
	P21/RXD	3	RXD	入力	CH1-SCI3(RS レベル)

デバッガ関係　J14　　適用ケーブル　AHC-E8(改造)					
	ﾎﾟｰﾄｼﾝﾎﾞﾙ CPU 基板	ピン番号 CPU 基板		方向	信号名
	/NMI	1		入力	強制ブレーク入力
	/RES	2		入力	リセット入力
	P87	3		入力	同期 CLK
	P86	4		出力	データ送信
	P85	5		入力	データ受信
	P24	6	未使用		
	P23	7	未使用		
	P22/TXD	8	未使用		
	P21/RXD	9	未使用		
	P20/SCK3	10	未使用		

モータドライブ関係　J2					
	ﾎﾟｰﾄｼﾝﾎﾞﾙ CPU 基板	ピン番号 CPU 基板		方向	信号名
PDR6	VCC	1		VCC	+5V
	P67	2	I/O	出力	LED1　　　0＝点灯
	P66	3	I/O	”	LED0　　　0＝点灯
	P65	4	FTIOB1	”	PWM1（サーボ）
	P64	5	FTIOA1	”	PWM2（右モータ）
	P63	6	I/O	”	右モータ方向　0＝正
	P62	7	I/O	”	左モータ方向　0＝正
	P61	8	FTIOB0	”	PWM3（左モータ）
	P60	9	I/O	入力	PushSW　　0＝ON
	GND	10		GND	GND

デバッガ関係　J13　　適用ケーブル AHC-E8(改造)					
	ﾎﾟｰﾄｼﾝﾎﾞﾙ CPU 基板	ピン番号 CPU基板		方向	信号名
	Vcc	1		VCC	+5V(電源供給のため)
	NC	2	未使用		
	P76/TMOV	3	未使用		
	P75/TMCIV	4	未使用		
	P74/TMRIV	5	未使用		
	NC	6	未使用		
	P72/TXD_2	7	未使用		
	P71/RXD_2	8	未使用		
	P70/SCK3_2	9	未使用		
	GND	10		GND	GND(電源供給のため)

＜AHC-E8(改造)　結線図＞

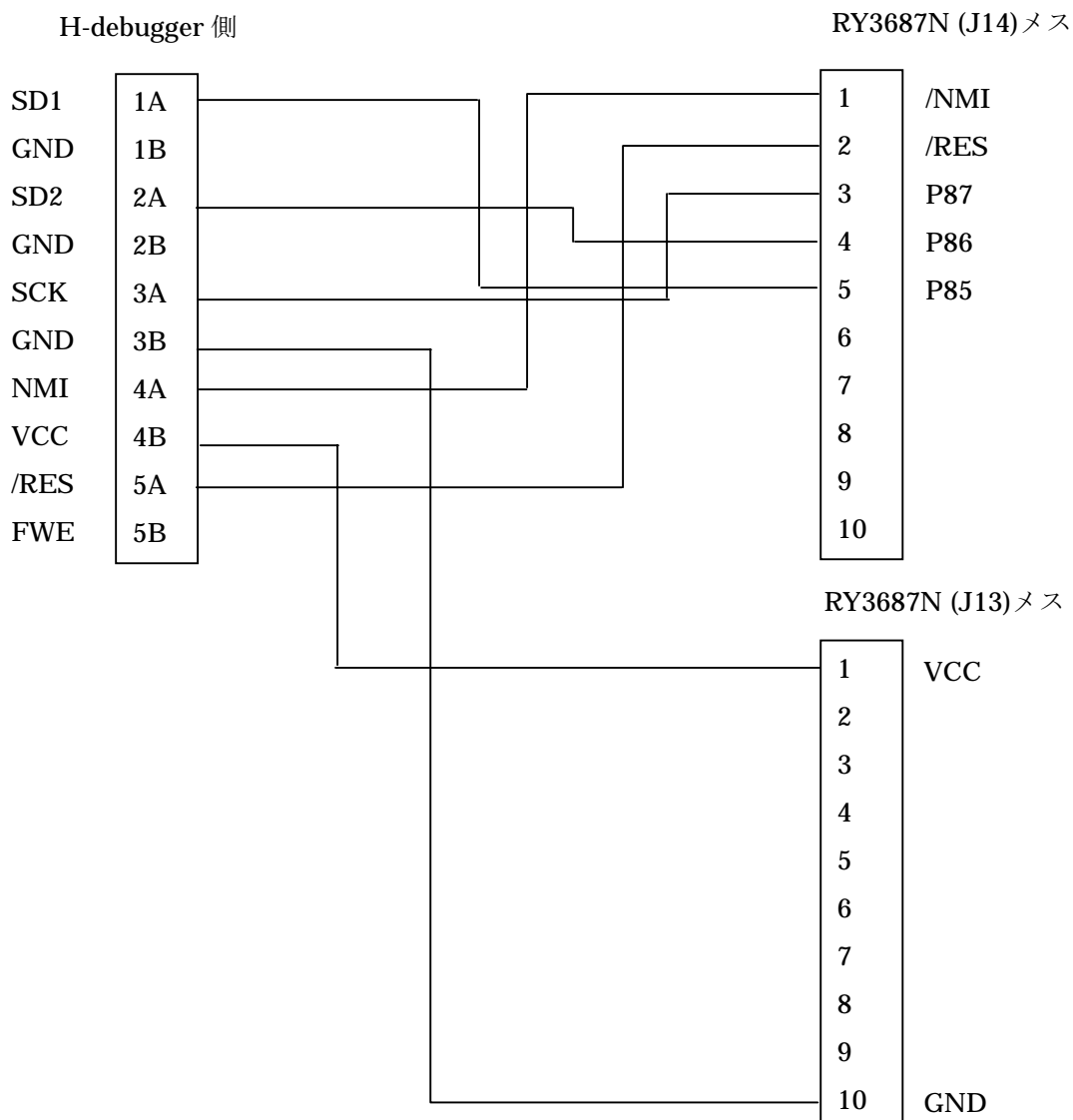
H-debugger 側

信号	ピン
SD1	1A
GND	1B
SD2	2A
GND	2B
SCK	3A
GND	3B
NMI	4A
VCC	4B
/RES	5A
FWE	5B

RY3687N (J14)メス

ピン	信号
1	/NMI
2	/RES
3	P87
4	P86
5	P85
6	
7	
8	
9	
10	

RY3687N (J13)メス

ピン	信号
1	VCC
2	
3	
4	
5	
6	
7	
8	
9	
10	GND

第５章　サンプルアプリケーションの準備と動作確認

５－１．Hew での準備（ルネサス製）

１）TOPPERS のホームページより、OSEK カーネル最新リリースをダウンロードして下さい。

URL:http://www.toppers.jp/osek-os.html

２）ダウンロードした「osek_os-x.x.lzh」を、適当なディレクトリに置き解凍して下さい。

３）サンプルアプリケーション用 CD の「.¥toppers_osek_Kpit_3687」の指定ファイルを、OSEK カーネルにコピーします。

＜CD 側＞ ＜DL した OSEK 側＞

.¥toppers_osek_Kpit_3687¥config¥h8t-kpitgnu-3687 --> .¥toppers_osek¥config
.¥toppers_osek_Kpit_3687¥syslib¥h8t-kpitgnu-3687 --> .¥toppers_osek¥syslib
.¥toppers_osek_Kpit_3687¥tools¥h8t-kpitgnu-3687 --> .¥toppers_osek¥tools

上記ディレクトリ下の全ファイルを、OSEK カーネルにディレクトリごと全コピーして下さい。

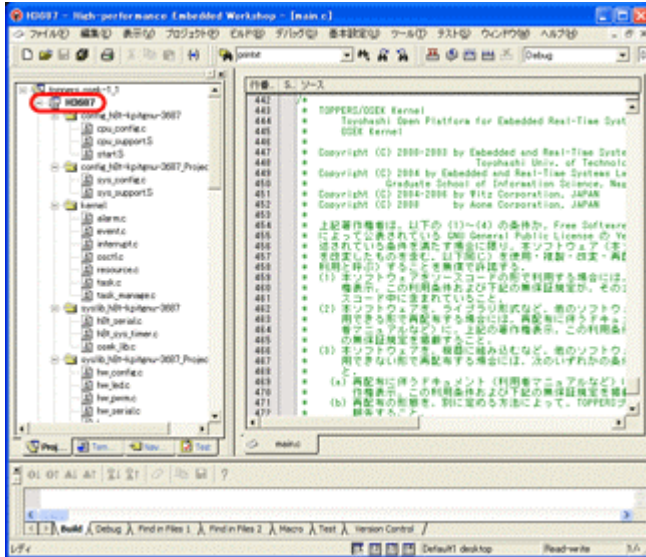
４）Hew を起動します。

・Hew Version 4.04.01.001―――――――――――――――――Renesas 製
・KPIT GNUH8[ELF] Toolchain v0801 ―――――――――――KPIT 製

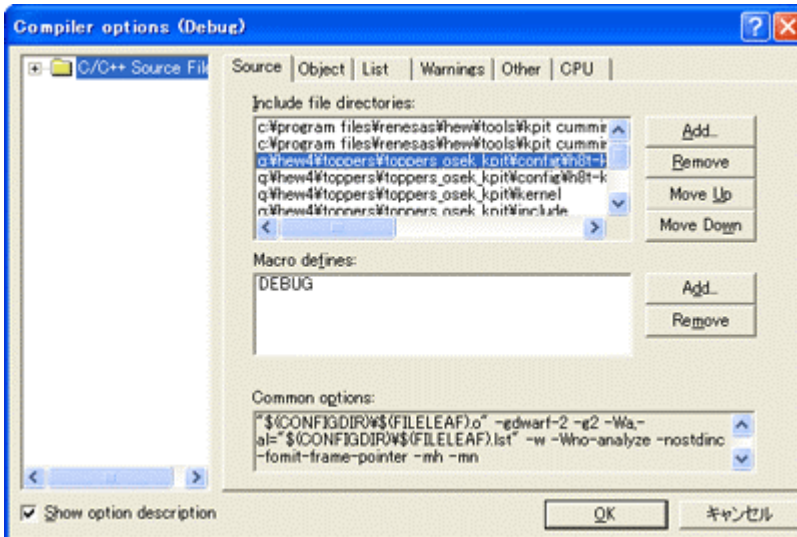
５）Hew メニューの＜ファイル＞－＜ワークスペースを開く＞でワークスペースを開きます。

・".¥toppers_osek¥tools¥h8t-kpitgnu-3687¥h8t_kpitgnu_3687.hws"を指定します。
・ディレクトリ情報が変わりますので、下記ウォーニングが表示されますが、気にせず「はい」を指定して下さい。

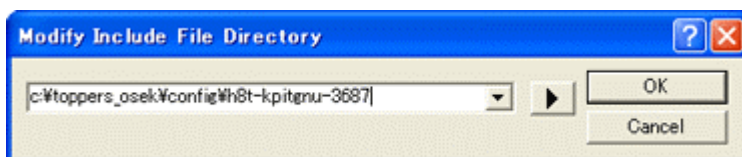
[5-1-1]

6）アクティブプロジェクトが「H3687」になっていることを確認します。H3687 になっていない場合は、Hew メニューの<プロジェクト>－<アクティブプロジェクトに設定>で「H3687」を指定して下さい。

[5-1-2]

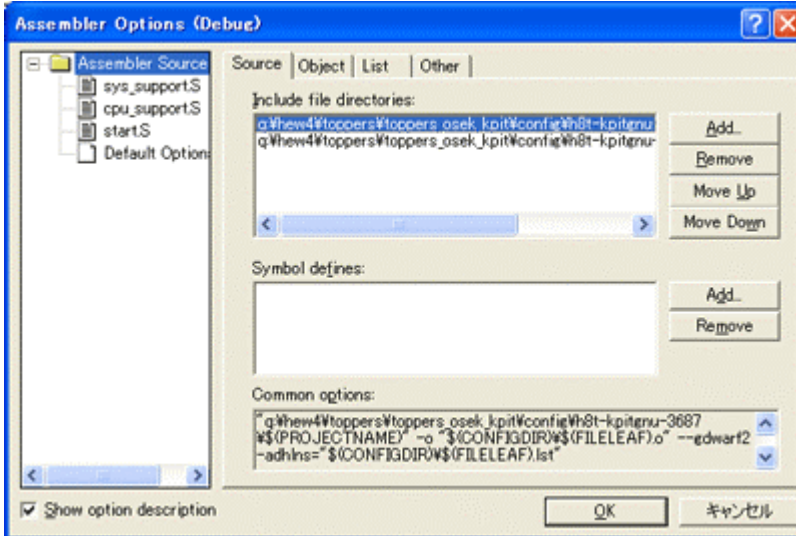
7）現 KPIT 版では、ディレクトリの相対指定[..¥]が出来ない為、インクルードディレクトリの変更が必要です。

Hew メニューの<ビルド>－<Compiler>をクリックして下さい。

[5-1-2-1]

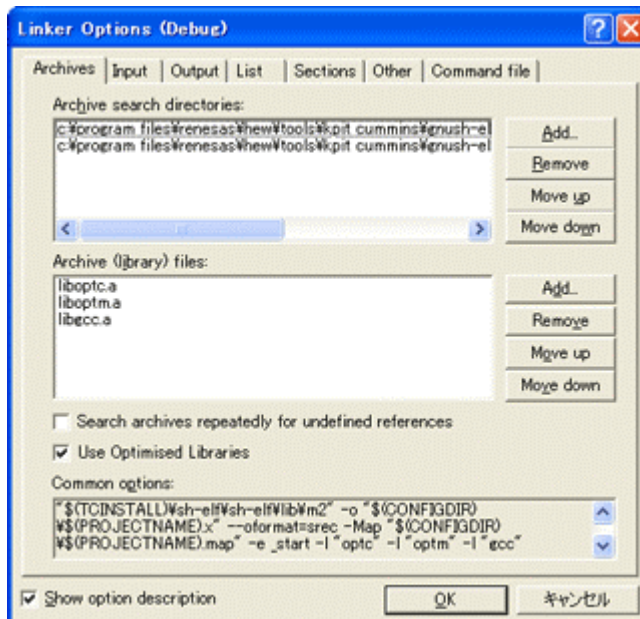
全てのディレクトリを現ワークスペースのディレクトリに変更して下さい。

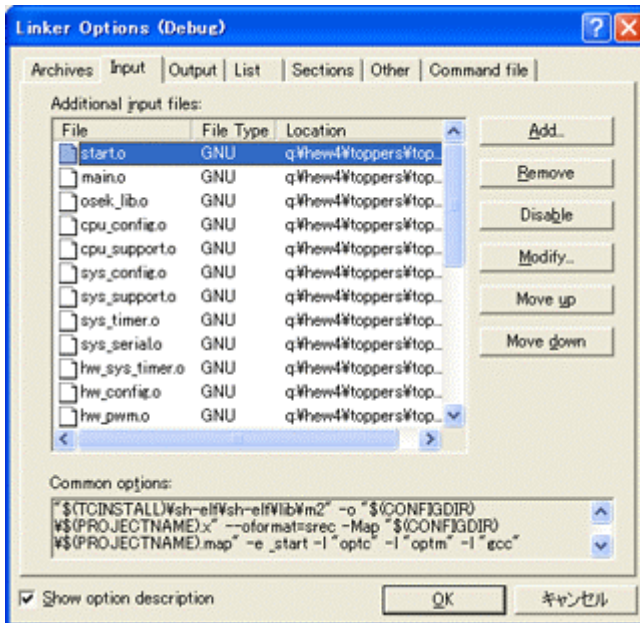
マウスでのダブリクリックで変更できます。

[5-1-2-2]

<Brows>指定が早いかと思います。

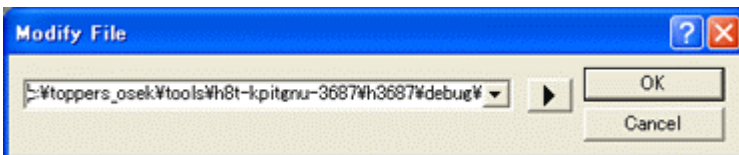
Hew メニューの＜ビルド＞－＜Assembler＞をクリックして下さい。

[5-1-2-3]

全てのディレクトリを現ワークスペースのディレクトリに変更して下さい。

Hew メニューの＜ビルド＞－＜Linker＞をクリックして下さい。

[5-1-2-4]

<Input>タグをクリックします。

[5-1-2-5]

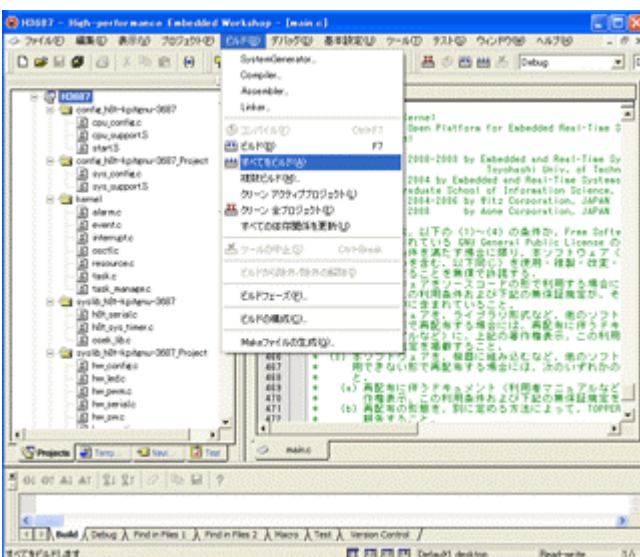
全てのディレクトリを現ワークスペースのディレクトリと相違があった場合は現ディレクトリに変更して下さい。
設定は、「$(CONFIGDIR)*.o」にしましたので変更する必要は無いかと思います。

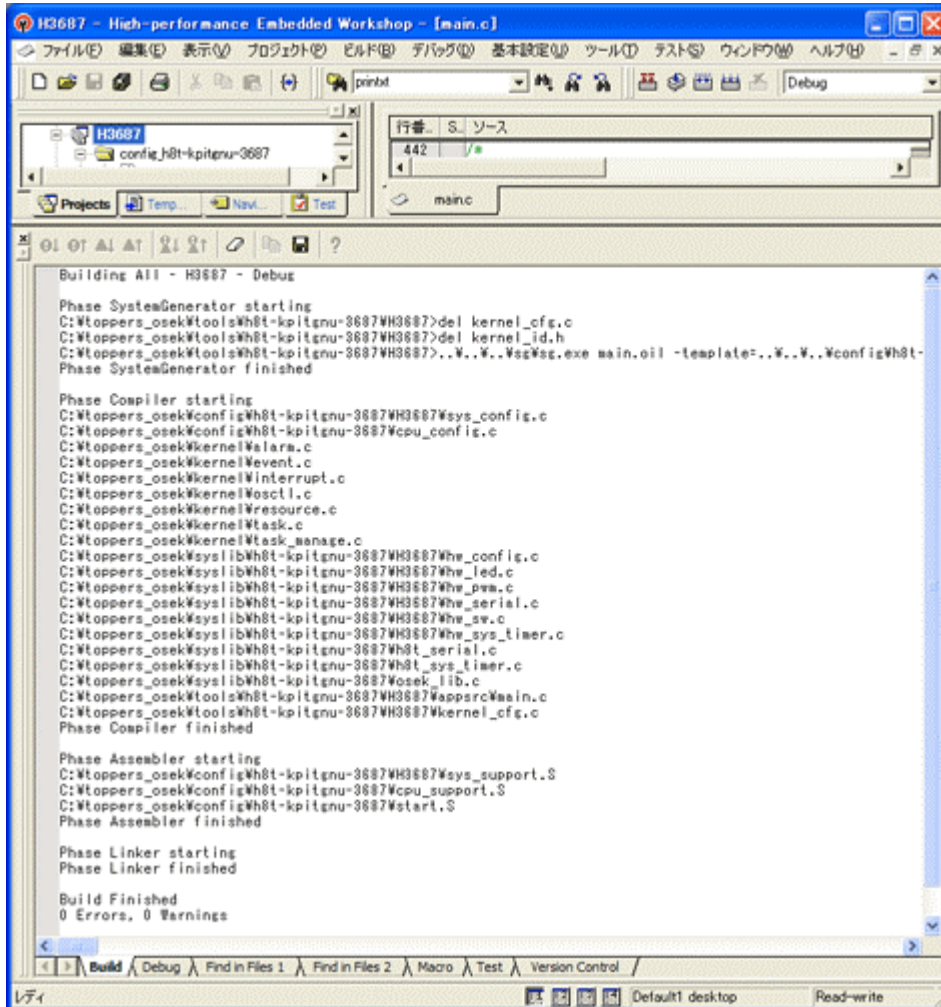
マウスでのダブリクリックで変更できます。

[5-1-2-6]

新規プロジェクトの作成時に必要です。

8）アクティブプロジェクトが「Project」に変更して、7）項と同じく<Compiler>と<Assembler>と<Linker>のディレクトリを変更と確認をして下さい。

9）アクティブプロジェクトを「H3687」に戻します

１０）Hew メニューの<ビルド>－<すべてをビルド>をクリックして下さい。

[5-1-3]

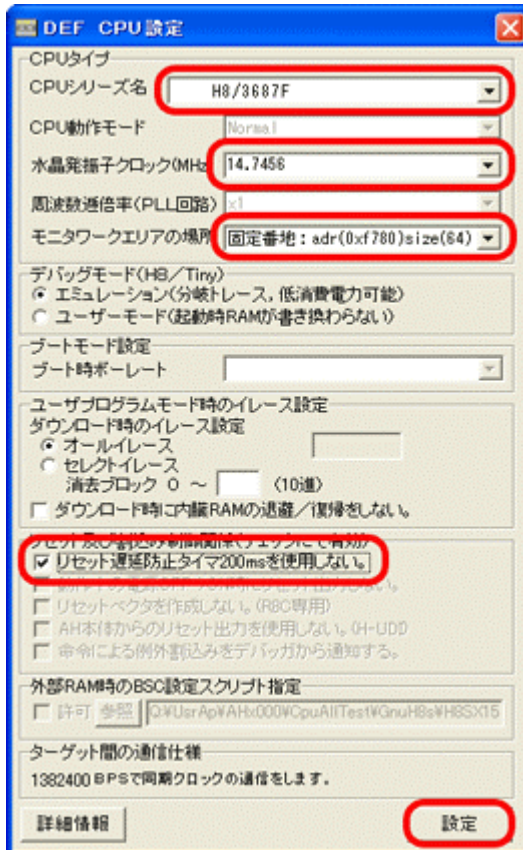
１２）ビルド結果

[5-1-4]

Build Finished「0 Errors 0 Warnings」になれば成功です。

５－２．H-debugger(DEF.exe)での準備（Aone 製）

１）H-debugger コントロールソフト「DEF.exe」を起動します。

２）DEF メニューの<オプション>－<環境設定>の「本体機種設定」が正しい機種と COM ポートの選択をされているか確認して下さい。

３）DEF メニュー<オプション>－<CPU 設定>をクリックします。

[5-2-1]

CPU シリーズ名　：H8/3687F

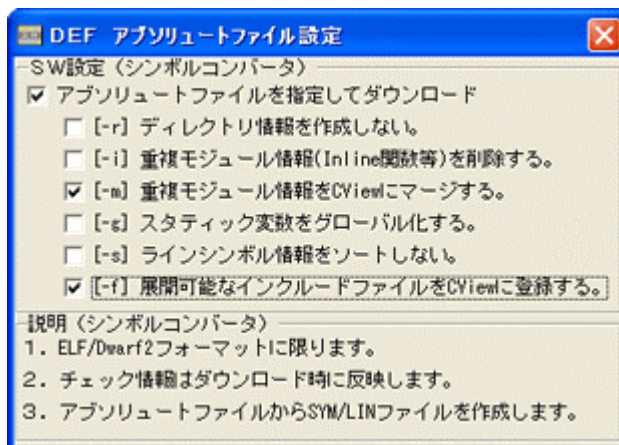
クロック(MHz)　：14.7456

ワークエリア　：固定番地方式

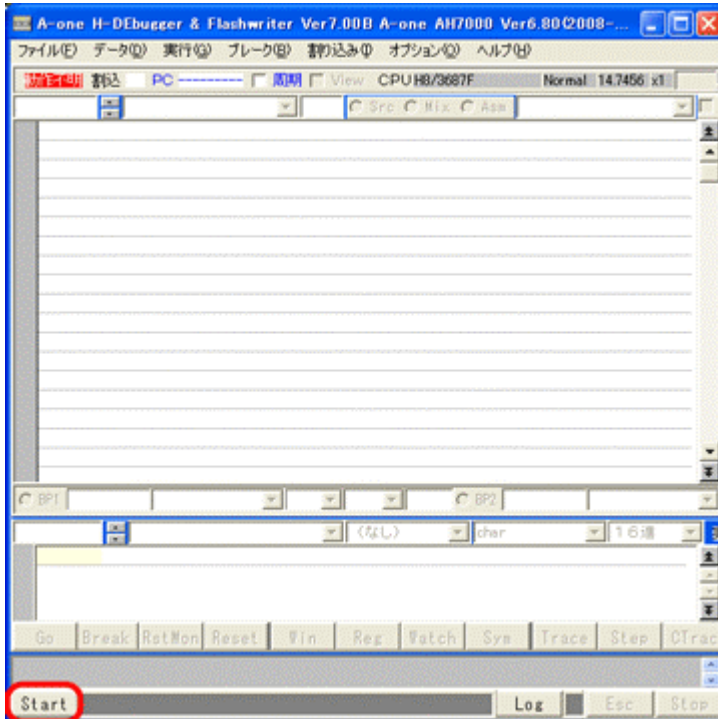
デバッグモード　：どちらでも良い

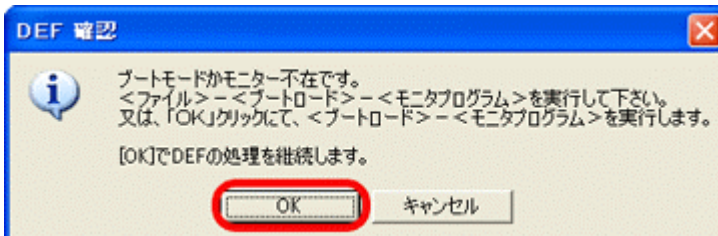
リセット遅延防止タイマ使用しない：チェック

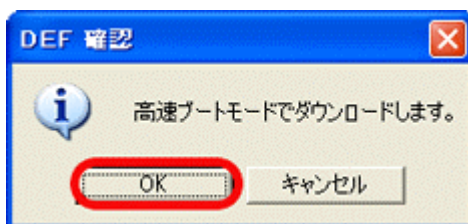
最低上記 4 項目を設定後、「設定」にクリックします。

４）DEF メニュー<ファイル>－<アブソリュートファイル設定>をクリックします。

[5-2-2]

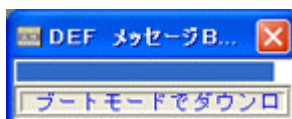
左図のように 3 箇所にチェックします。

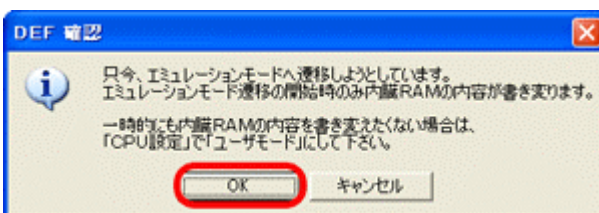
5）DEF 画面、左下隅の「Start」をクリックします。（ターゲット側の電源は ON の事）

[5-2-3]

6）最初だけモニタ不在の通知が表示されます。

[5-2-4]

「OK」をクリックします。

[5-2-4-1]

デバッグ用モニタを登録します。

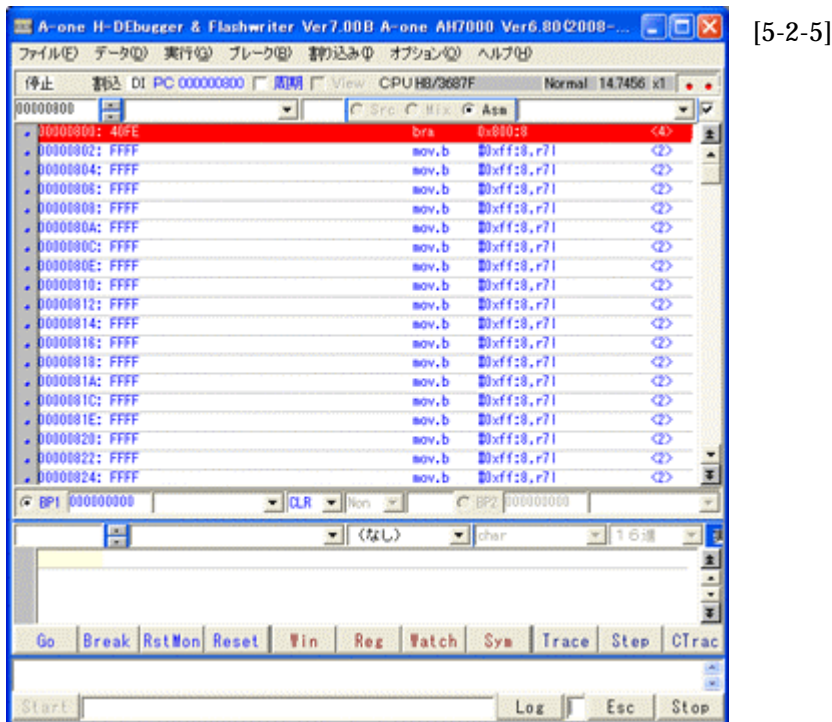
「OK」をクリックします。

[5-2-4-2]

モニタ書き込み中にインジケータが表示されます。

[5-2-4-3]

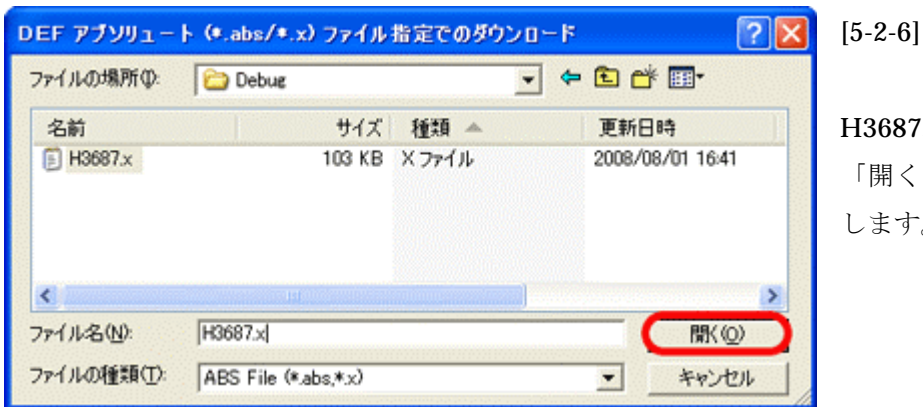
デバッグモードを「エミュレーションモード」に指定した場合のメッセージです。

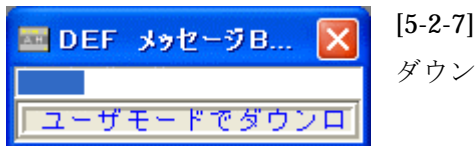
「OK」をクリックします。

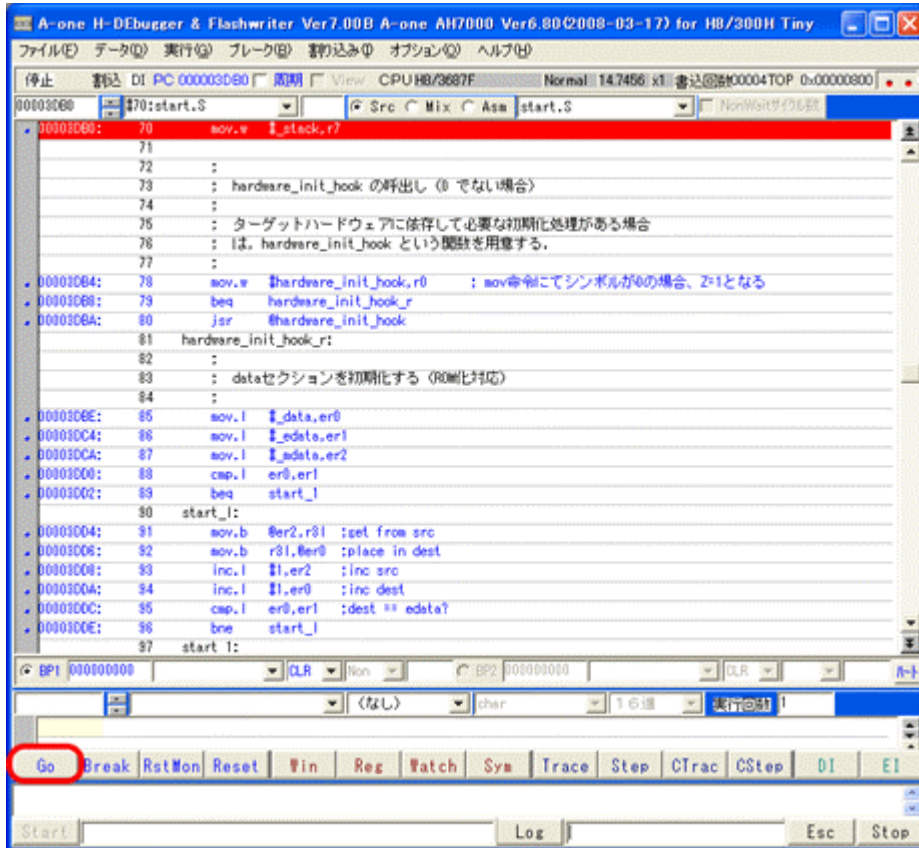
7）ターゲット側と正常な通信を確立しますと下図の様な画面になります。

[5-2-5]

8）DEF メニュー＜ファイル＞－＜ダウンロード＞をクリックします。

.¥toopers_osek¥tools¥h8t-kpitgnu-3687¥tools¥H3687¥debug　まで降ります。

[5-2-6]

H3687.x を選択後「開く」をクリックします。

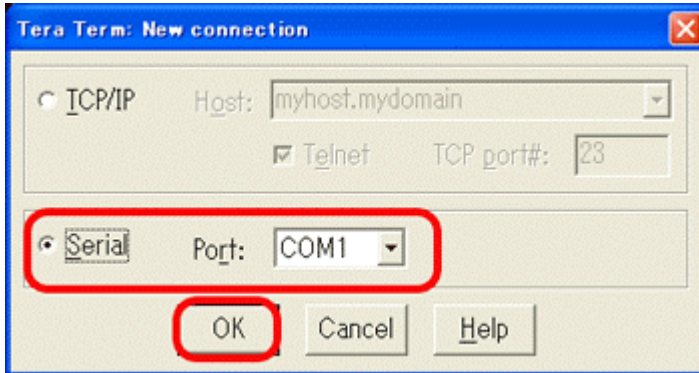
[5-2-7]

ダウンロード中は、このようなインジケータ表示します。

９）ダウンロードが成功しますと、下図のような DEF 画面になります。

[5-2-8]

ShortPB[RstMon]On でこの View 画面になります

5－3．サンプルアプリケーションを走らせる前の準備

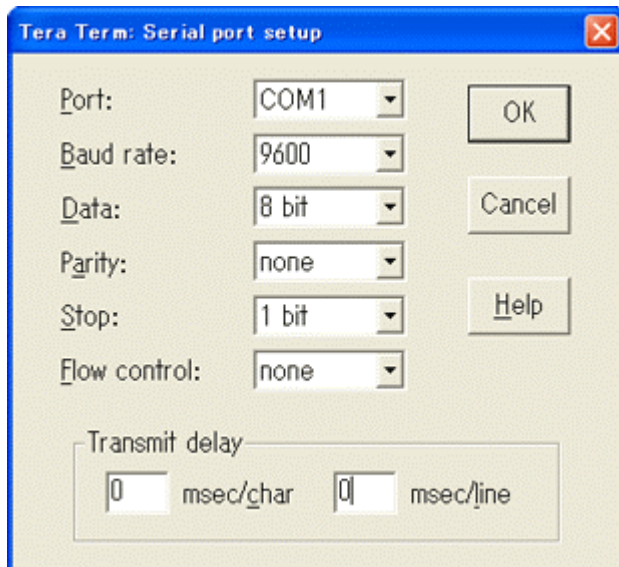
1）CPU 基板「RY3687N」上の DIP-SW(1)(2)(3)(4)全てを OFF にします。

2）パソコン側のソフト「Tera Term」を起動します。

[5-3-1]

「Serial」にチェックを入れ、使用パソコンのシリアル COM 番号を指定します。

「OK」をクリックします。

3）「Tera Term」メニューの＜Setup＞－＜Serial port＞をクリックします。

Tera Term: Serial port setup
Port: COM1
Baud rate: 9600
Data: 8 bit
Parity: none
Stop: 1 bit
Flow control: none
Transmit delay
0 msec/char
0 msec/line
OK
Cancel
Help

[5-3-2]

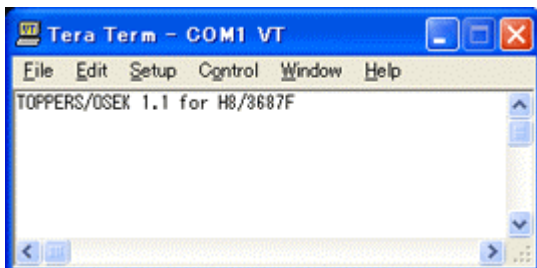
Port 以外の通信仕様を左図のように設定して下さい。

「OK」をクリックします。

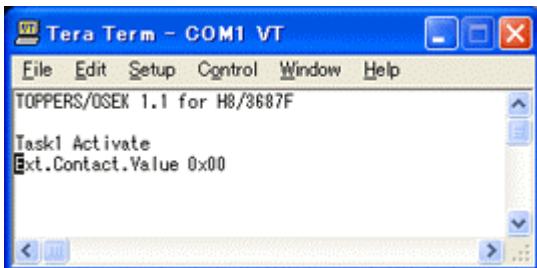
５－４．サンプルアプリケーションを走らせます。

１）DEF のショート PB「Go」をクリックします。[5-2-8]図を参照

「Tera Term」画面は下画面のようになります。

[5-4-1]

２）センサン入力タスク(Task1)を起動します。

①CPU 基板「RY3687N」上の DIP-SW(1)を ON にします。

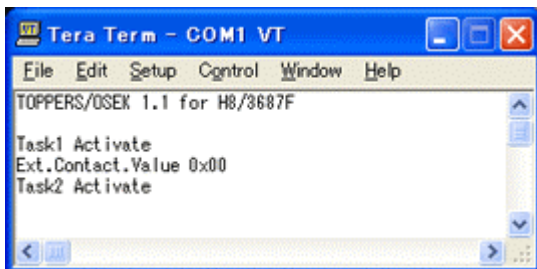
「Tera Term」画面は下画面のようになります。

[5-4-2]

センサ入力値を 16 進数で表示します。

３）PWM 出力タスク(Task2)を起動します。

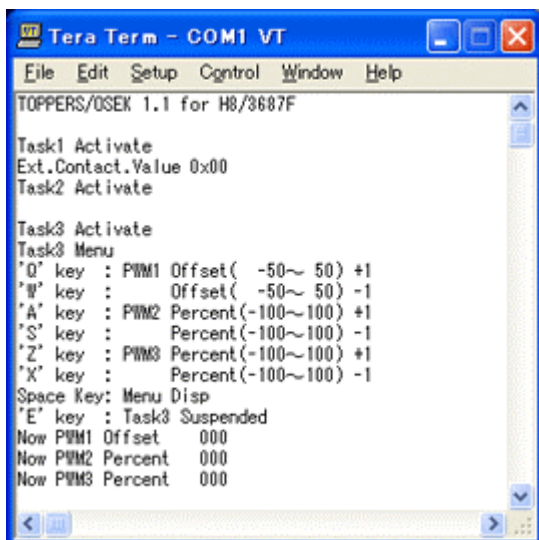
①モータドライブ基板側の電源「DC+6V」を供給します。

②モータドライブ基板の PushSW を ON します。

「Tera Term」画面は下画面のようになります。

[5-4-3]

4）RS 設定タスク(Task3)を起動します。

①「Tera Term」よりスペースキーを入力します。

「Tera Term」画面は下画面のようになります。

[5-4-4]

②PWM1（サーボ）を出力します。

Q と W　キーを押して下さい。数値にもとずいてサーボが動作します。

③PWM2（右モータ）を出力します。

A と S　キーを押して下さい。数値にもとずいて右モータが動作します。

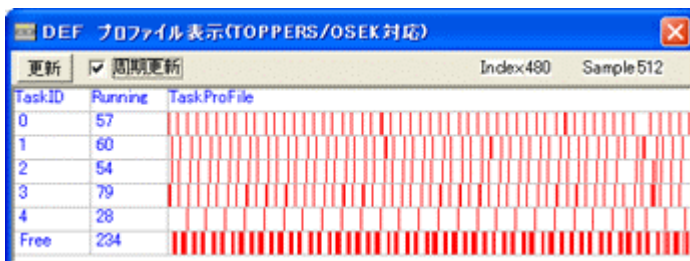
＋数値は正転、－数値は逆転します。

④PWM3（左モータ）を出力します。

Z と X　キーを押して下さい。数値にもとずいて左モータが動作します。

＋数値は正転、－数値は逆転します。

5）各タスクの Running の状態を見てみます。

①DEF メニュー<データ>－<プロファイル表示>をクリックします。

[5-4-5]

各タスクの Run 状態を見ることができます。

6）作者より

本サンプルの利用により、TOPPERS/OSEK カーネル下でのユーザーアプリケーションプログラムの作成に少しでもお役に立てれば幸いです。

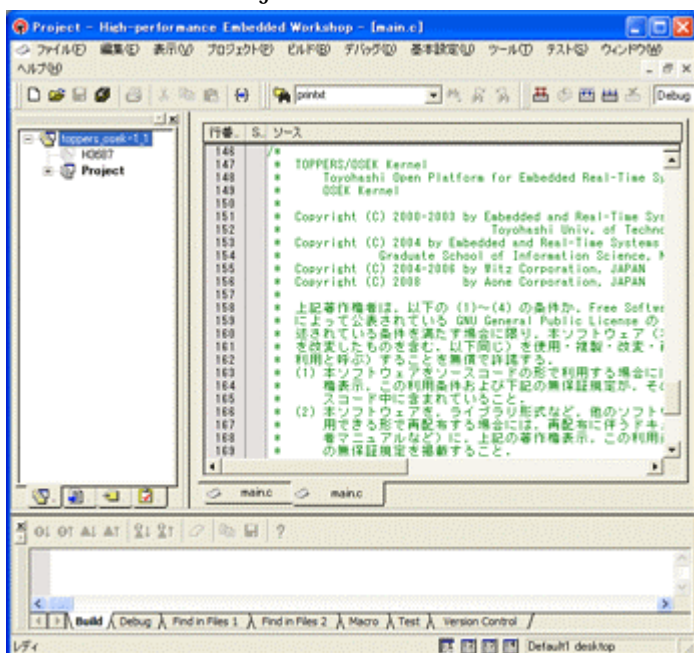
第６章　新規プロジェクトを追加する場合の手順例

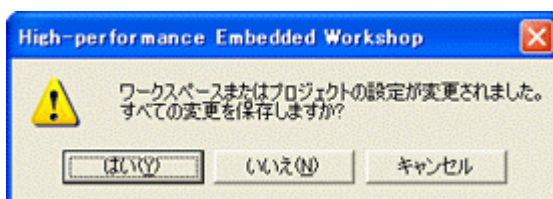
６－１．プロジェクトタイプの作成

TOPPERS/OSEK 下で新規プロジェクトを追加する場合、Hew 設定を簡略化するための手順案を記述します。

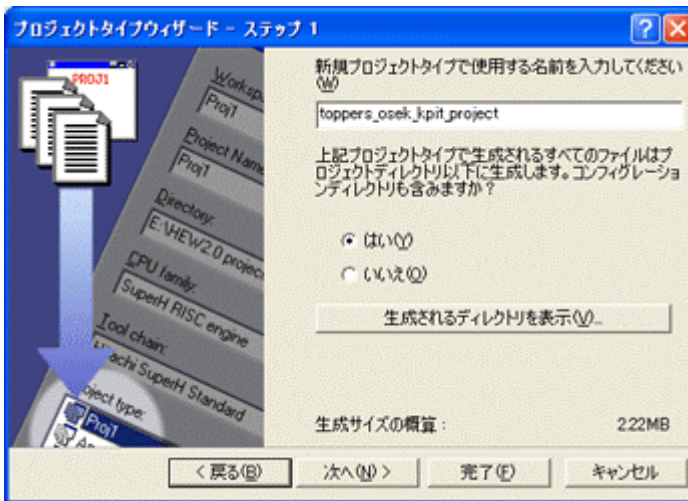
⚠ H3687 用のプロジェクトテンプレーを作成しましたので、下記手順でプロジェクトタイプ（カスタム）を作成して下さい。

（必ず Compiler]と Assembler と Linker のディレクトリが変更済みである事！！）

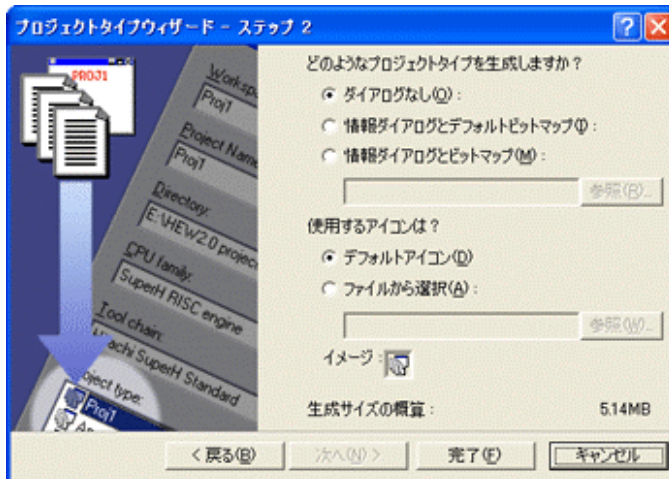
１）空プロジェクト「Project」をアクティブプロジェクトに指定します。

[6-1-1]

２）Hew メニュー＜プロジェクト＞－＜プロジェクトタイプの作成＞をクリックします。

[6-1-2]

保存を促すメッセージです。

「はい」をクリックします。

[6-1-3]
プロジェクト新規登録時に指定するタイプ名を入力する。
ex)toppers_osek_kpit_project

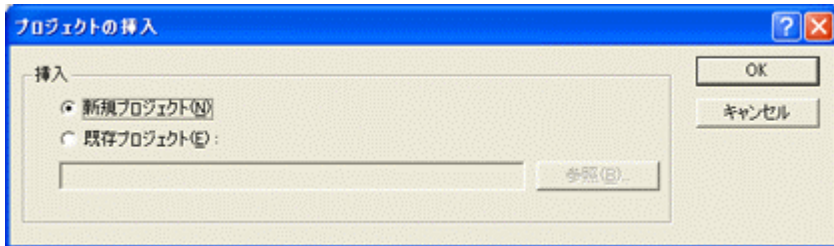
はい(Y)側をチェック

「次へ」をクリックします。

[6-1-4]

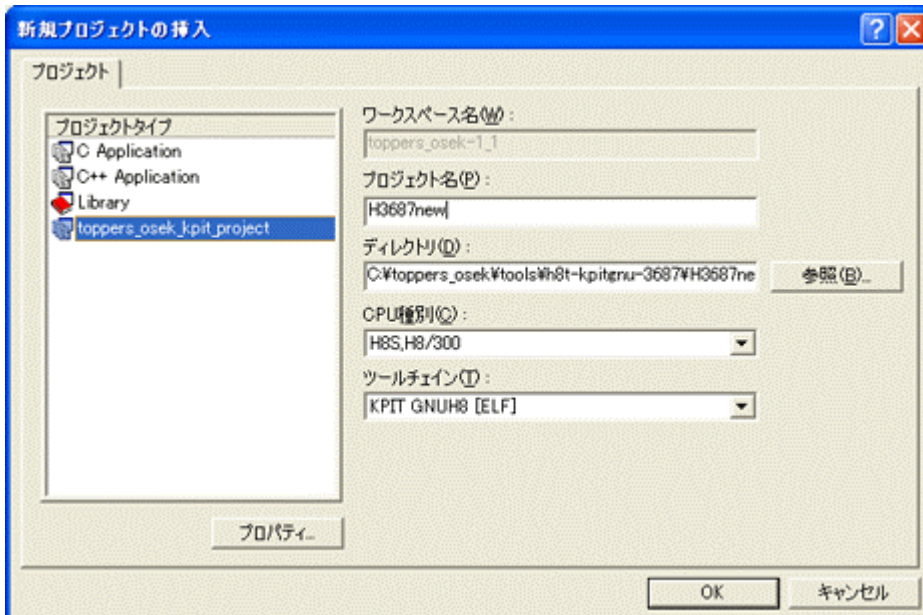
左図の様に「デフォルト」のままで、

「完了」をクリックします。

6－2．新規プロジェクトを登録します。

1） Hew メニュー<プロジェクト>－<プロジェクトの挿入>をクリックします。

[6-2-1]

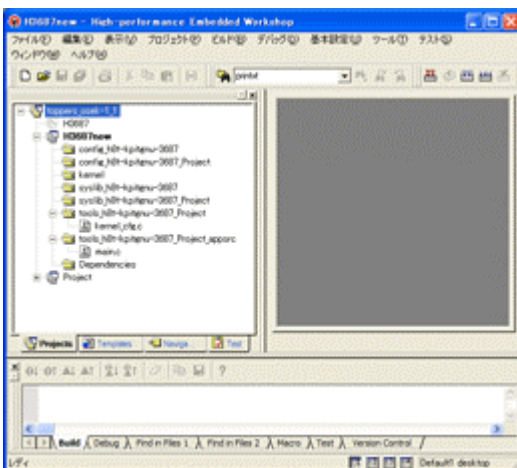
新規プロジェクト側チェックにて「ＯＫ」をクリックします。

2）プロジェクト名を登録します。

[6-2-2]

①プロジェクトタイプを前項で登録した「toppers_osek_kpit_project」を指定します。

②プロジェクト名に任意な目的プロジェクト名を入力します。後の説明でプロジェクト名が必要になりますので、ここでは新規プロジェクト名を「**H3687new**」とします。

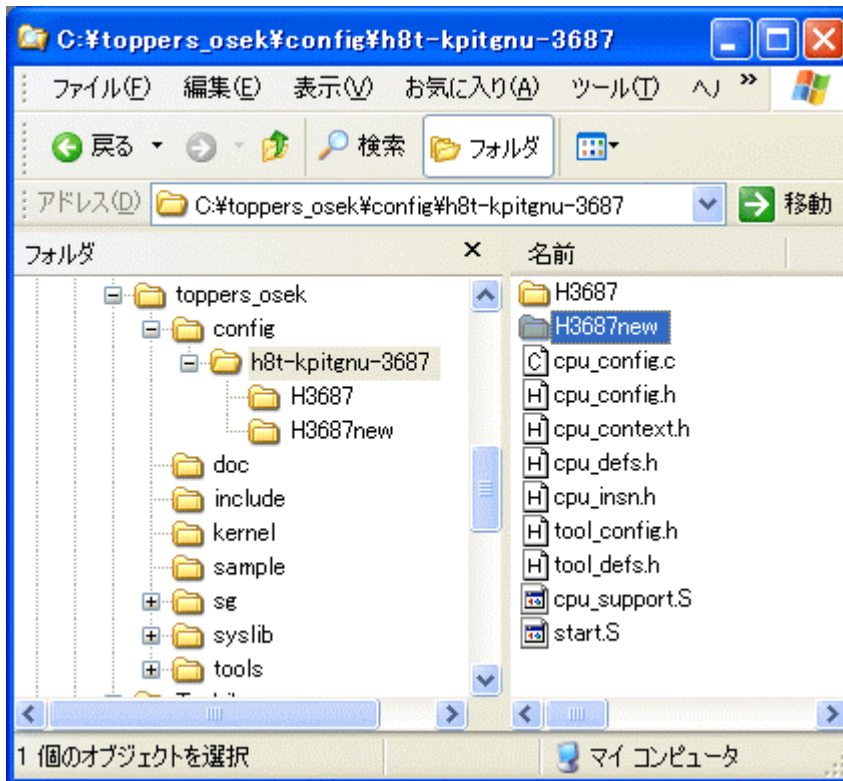
③上記設定で「ＯＫ」をクリックします。

3） Hew に新規プロジェクトが作成されます。

[6-2-3]

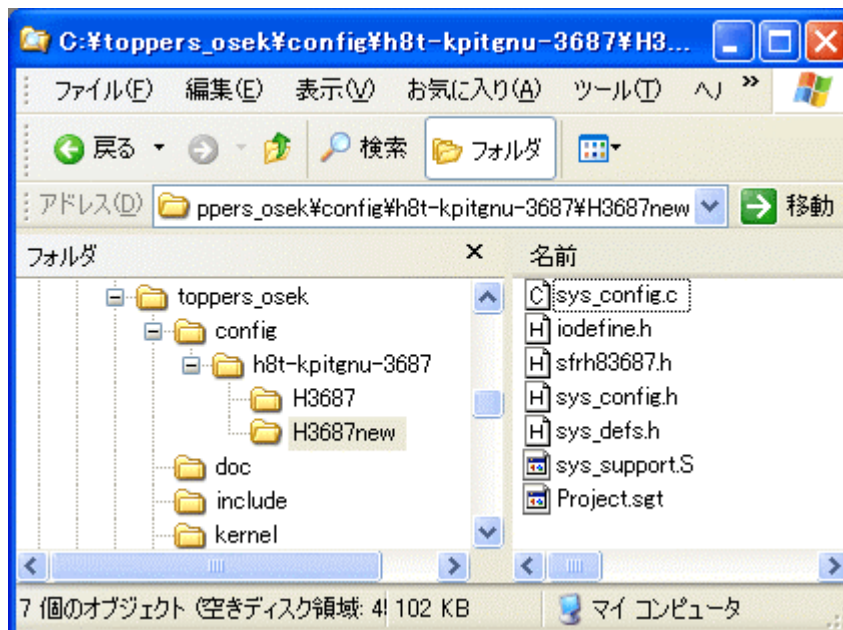
左図のように、新規プロジェクトが Hew に登録されます。

4）新規プロジェクト用に新規ディレクトリ作成とファイルをコピーします。

① .¥toopers_osek¥**config**¥h8t-kpitgnu-3687 の下に、

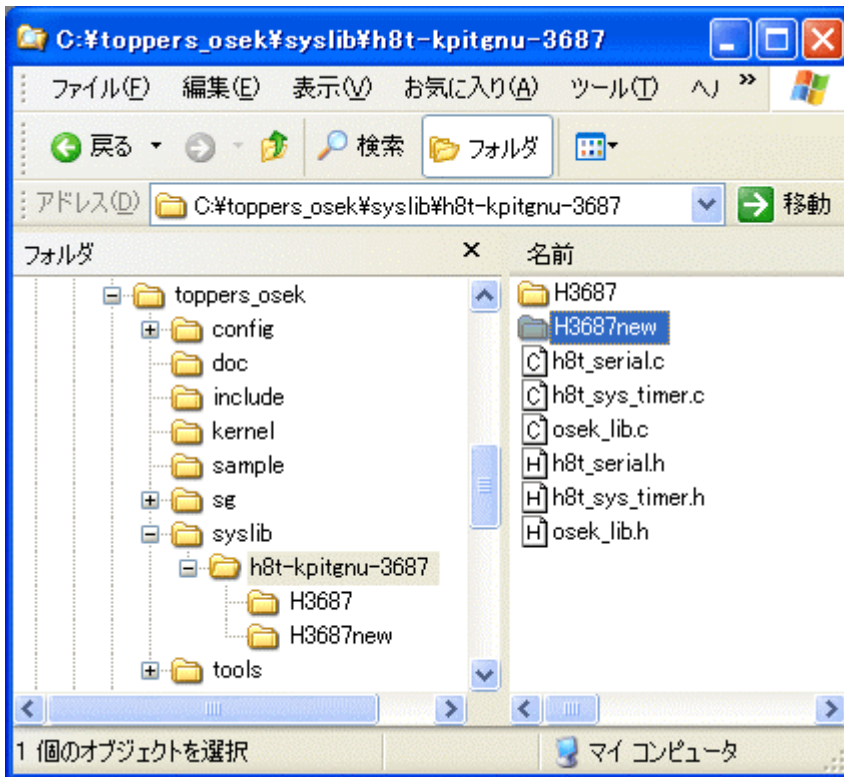
新規プロジェクト名「**H3687new**」のディレクトリを作成します。

[6-2-4-1]

＜追加 1 ＞

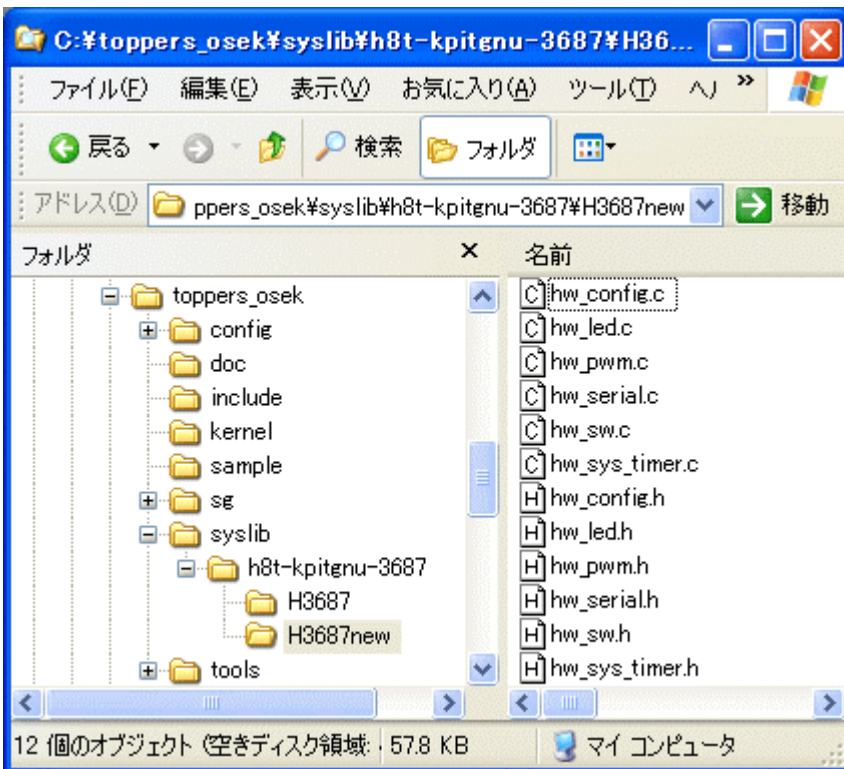
②作成した「**H3687new**」に、「H3687」下の全ファイルをコピーします。

[6-2-4-2]

コピー確認

① .¥toopers_osek¥syslib¥h8t-kpitgnu-3687 の下に、

新規プロジェクト名「H3687new」のディレクトリを作成します。

[6-2-4-3]

＜追加２＞

②作成した「H3687new」に、「H3687」下の全ファイルをコピーします。

[6-2-4-4]

コピー確認

5）システムジェネレータ用バッチファイルの一部を変更します。

.¥toopers_osek¥tools¥h8t-kpitgnu-3687¥H3687new¥call_sg.bat

のファイルを何らかのエディタで開きます。

＜元ファイル＞ [6-2-5-1]

```
@REM SG 実行バッチファイル

@REM カーネルコンフィグレーション
del kernel_cfg.c
del kernel_id.h
..¥..¥..¥sg¥sg.exe   main.oil
-template=..¥..¥..¥config¥h8t-kpitgnu-3687¥Project¥Project.sgt
-I..¥..¥..¥sg¥impl_oil -os=ECC2
```

＜変更ファイル＞ [6-2-5-2]

```
@REM SG 実行バッチファイル

@REM カーネルコンフィグレーション
del kernel_cfg.c
del kernel_id.h
..¥..¥..¥sg¥sg.exe   main.oil
-template=..¥..¥..¥config¥h8t-kpitgnu-3687¥H3687new¥Project.sgt
-I..¥..¥..¥sg¥impl_oil -os=ECC2
```

上記の様に、Project を新規プロジェクト名「H3687new」に変更します。

6）新規プロジェクト「H3687new」にユニット「ソースファイル」を登録します。

①フォルダ名「config_h8t-kpitgnu-3687」に登録

Hew メニュー＜プロジェクト＞－＜ファイルの追加＞をクリックします。

.¥toppers_osek¥config¥h8t-kpitgnu-3687 に移動します。

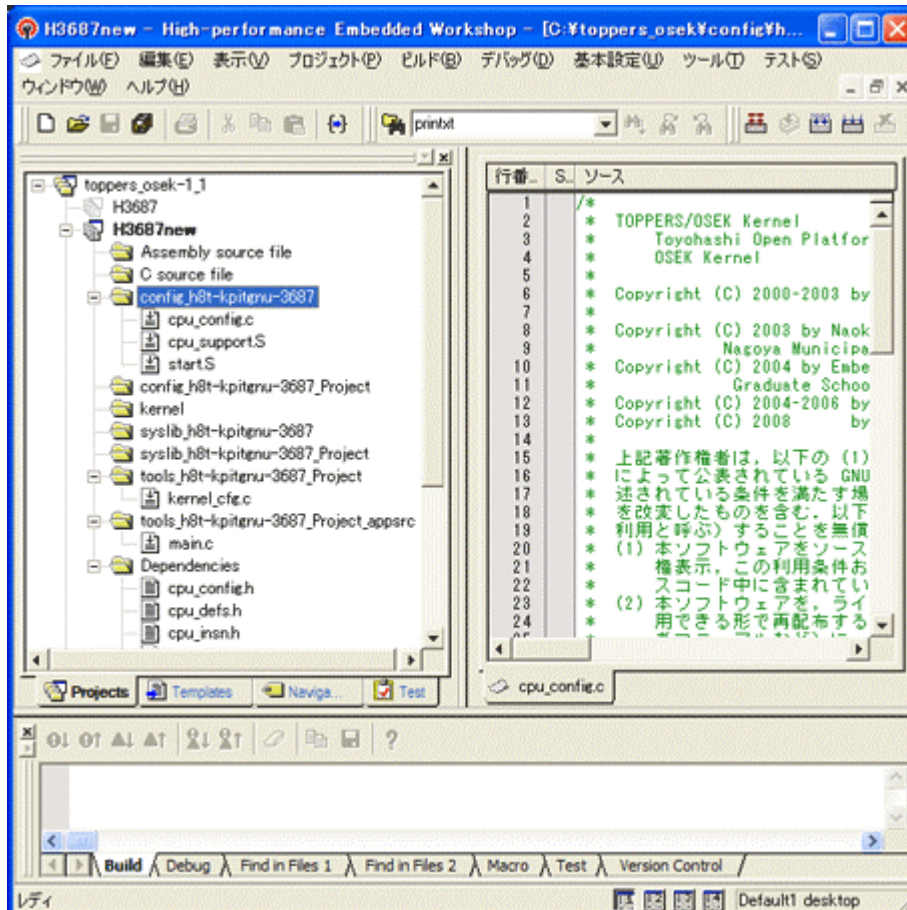
[6-2-6-1]

①cpu_config.c

②cpu_support.s

③start.s

の 3 ファイルを「追加」します。

ファイル選択後「追加」をクリックしますと、Hew ツリーでは一旦「Assembly souce file」と「C souce file」のフォルダに入りますので、マウスドロップにて目的フォルダに移動して下さい。

[6-2-6-2]

移動後の状態

②フォルダ名「config_h8t-kpitgnu-3687_Project」に登録

Hew メニュー<プロジェクト>－<ファイルの追加>をクリックします。

.¥toppers_osek¥config¥h8t-kpitgnu-3687¥H3687new に移動します。

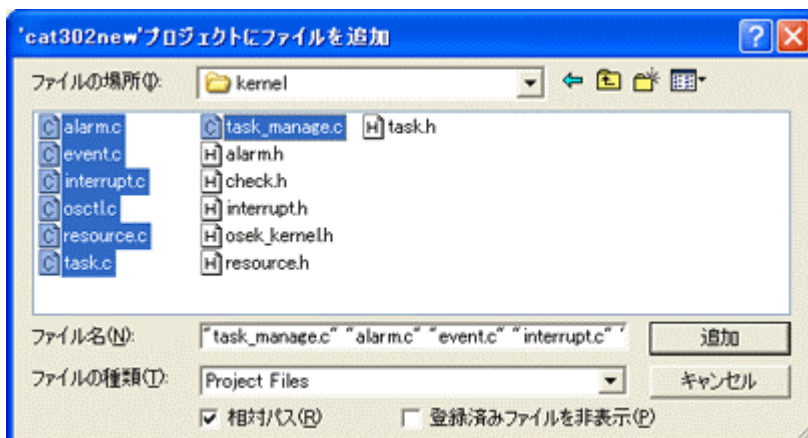
[6-2-6-3]
①sys_config.c
②sys_suport.s

の 2 ファイルを「追加」します。

前項と同じ様に、別フォルダに入りますので、マウスドロップで目的フォルダに移動して下さい。

③フォルダ名「kernel」に登録

Hew メニュー<プロジェクト>－<ファイルの追加>をクリックします。

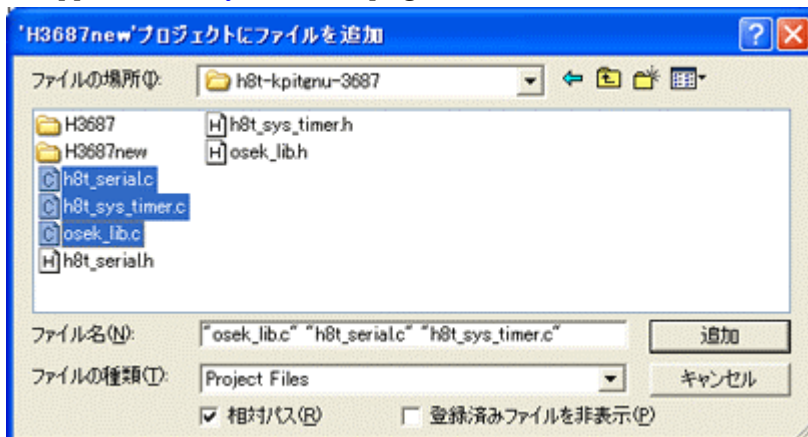
.¥toppers_osek¥kernel に移動します。

[6-2-6-4]
①alarm.c
②event.c
③interrupt.c
④osctl.c
⑤resource.c
⑥task.c
⑦task_manage.c

の 7 ファイルを「追加」します。

前項と同じ様に、別フォルダに入りますので、マウスドロップで目的フォルダに移動して下さい。

④フォルダ名「syslib_h8t-kpitgnu-3687」に登録

Hew メニュー＜プロジェクト＞－＜ファイルの追加＞をクリックします。

.¥toppers_osek¥syslib¥h8t-kpitgnu-3687 に移動します。

[6-2-6-5]

①sys_timer.c

②sys_serial.c

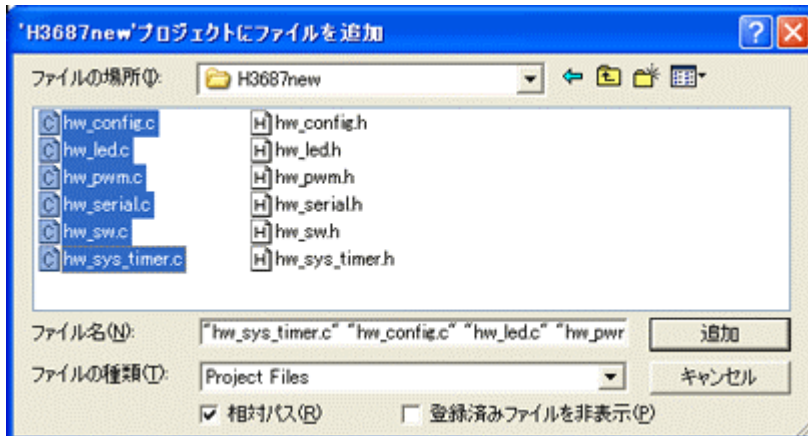
③osek_lib.c

の 3 ファイルを「追加」します。

前項と同じ様に、別フォルダに入りますので、マウスドロップで目的フォルダに移動して下さい。

⑤フォルダ名「syslib_h8t-kpitgnu-3687_Project」に登録

Hew メニュー＜プロジェクト＞－＜ファイルの追加＞をクリックします。

.¥toppers_osek¥syslib¥h8t-kpitgnu-3687¥H3687new に移動します。

[6-2-6-6]

①hw_config.c

②hw_led.c

③hw_pwm.c

④hw_serial.c

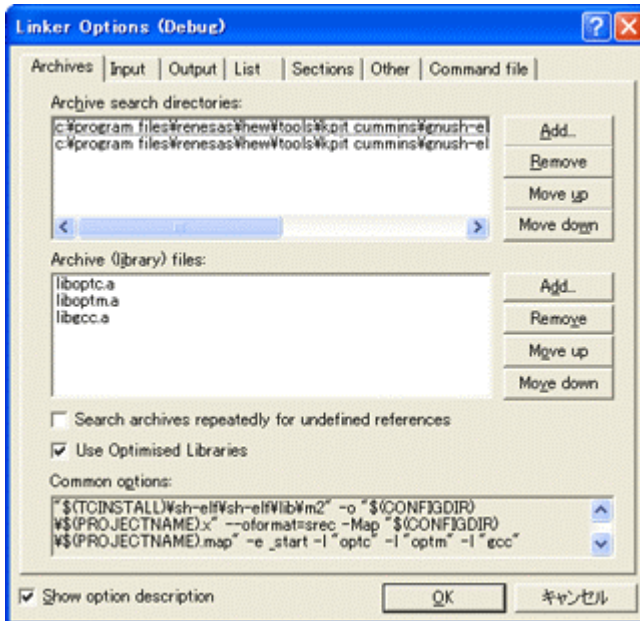
⑤hw_sw.c

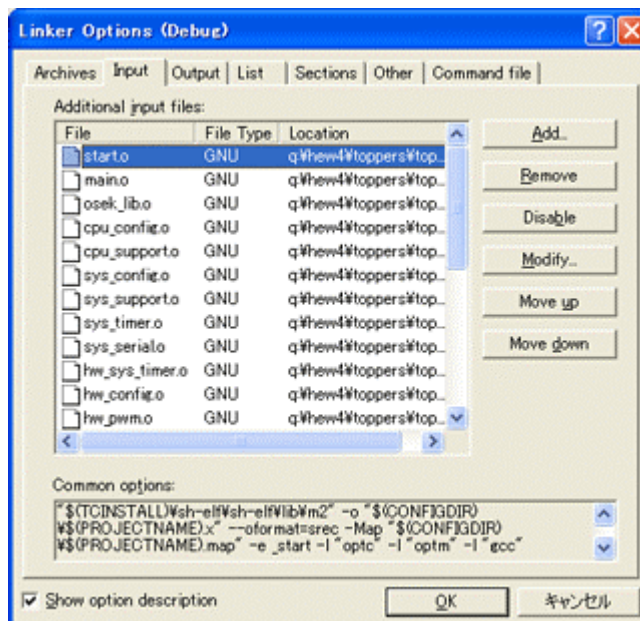
⑥hw_sys_timer.c

の 6 ファイルを「追加」します。

前項と同じ様に、別フォルダに入りますので、マウスドロップで目的フォルダに移動して下さい。

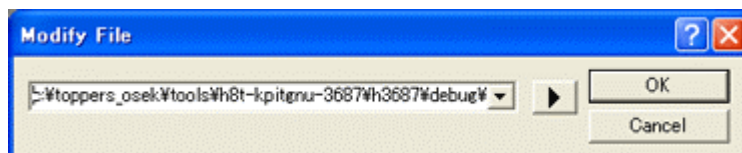
7）新規プロジェクト「H3687new」のリンク順番を指定します。

Hew メニューの＜ビルド＞－＜Linker＞をクリックして下さい。

[6-2-7-1]

〈Input〉タグをクリックします。

[6-2-7-2]

全てのディレクトリを現ワークスペースのディレクトリと相違があった場合は現ディレクトリに変更して下さい。

設定は、「$(CONFIGDIR)*.o」にしましたので変更する必要は無いかと思います。

マウスでのダブリクリックで変更できます。

[6-2-7-3]

変更が必要な場合は、「h3687new」のディレクトリに変更する。

⚠ 以上の作業で、新規プロジェクトの追加作業は終了です。目的のプロジェクト仕様に合わせた「main.c」を作成して下さい。
フォルダ名「tools_h8t-kpitgnu-3687_Project_appsrc」
ディレクトリ「.¥toppers_osek¥tools¥h8t-kpitgnu-3687¥H3687new¥appsrc」に空ファイルとして用意してあります。

８）OS 定義および Task の追加やプライオリティを変更したい場合は、ディレクトリ「.¥toppers_osek¥tools¥h8t-kpitgnu-3687¥H3687new」にある、「main.oil」のテキストファイルを変更して下さい。

定義仕様に関しては、「OPPERS/OSEK カーネル SG 取扱説明書」株式会社ヴィッツ製をご覧下さい。

第７章　備考

７－１．おわりに

本アプリケーションノートは、いたらない所が多々有ると思います。意味不明な箇所がありましたら、遠慮なくメールにて申し付け下さい。積極的に改訂し、より判り易いノートにしたく思っていますので、皆様の御協力を御願い申し上げます。

2008 年 8 月　著者

〒４８６－０８５２
愛知県春日井市下市場町 6-9-20
エーワン株式会社
Tel　　0568-85-8511
Fax　　0568-85-8501
E-mail cat-i@aone.co.jp
URL　　http://www.aone.co.jp